Differential Algebraic Description of Beam Dynamics to Very High Orders

M. Berz

SSC Central Design Group
Universities Research Association
c/o Lawrence Berkeley Laboratory
Berkeley, Ca 94720

Published in "Particle Accelerators",
Volume 24,
1989

Abstract

The new method of differential algebras for the description of beam dynamics is presented. It allows a very straightforward and elegant computation of transfer maps of accelerator systems which can be analysed for data relevant to accelerators. The order of the procedure is unlimited. The theoretical background of the method is presented in detail. It is shown how the method can be used in practice.

1 Introduction

The effect of an accelerator section can be described mathematically by a map relating the final coordinates $\vec{z}_f$ of a particle to the initial coordinates $\vec{z}_i$

$$\vec{z}_f = \mathcal{M}(\vec{z}_i, \vec{\delta}) \tag{1}$$

The coordinates $\vec{z}$ contain positions and momenta of the particle. The vector $\vec{\delta}$ contains other parameters that influence the motion like energy, mass or charge of the particle or accelerator parameters like certain multipole strengths. From this map $\mathcal{M}$, quantities of interest for accelerators like tune shifts and chromaticities can be extracted. This is described in detail in a companion paper [16].

Except for the most trivial cases it is impossible to find a closed analytical solution for the map $\mathcal{M}$. However, expanding $\mathcal{M}$ in a power series yields a set of differential equations for the expansion coefficients which in many cases can be solved analytically up to some order. The complexity of the resulting differential equations, however, increases dramatically with the order of the expansion coefficients. So this procedure is limited to low or medium orders. In fact, most widely used codes [1], [2], [3] only have the capability to compute nonlinearities in (1) through third order.

Recently it has been possible to extend this to higher orders using the custom made formula manipulator HAMILTON [4]. This program produces FORTRAN code for formulas of nonlinearities compatible with the program COSY [5]. At the present time, COSY can compute all fifth order nonlinearities of common beam line elements like bending magnets and magnetic and electrostatic multipoles including the dependence on the particle mass.

Since the accuracy of the Taylor series representation of the map in equation (1) increases by using higher orders, it is desirable to know $\mathcal{M}$ to as high an order as possible. This certainly holds for the purpose of using the map for subsequent symplectic tracking of simulation particles; but also for the above mentioned purely analytical purposes like computation of tune shifts with amplitude, chromaticities, invariants etc. based on Hamiltonian perturbation theory [6], [7], [16], this is highly desirable.

In this paper we will present a very straightforward way to compute nonlinearities to arbitrary orders based on differential algebraic techniques. The partial derivatives are computed to machine precision; the whole procedure is completely independent of the order and only limited by the power of the computer.

Besides the elegance in the computation of transfer maps, the method is so versatile that it allows the computation of arbitrary derivatives. Here no analytic formulas for derivatives have to be derived; on the other hand, the method is always accurate to machine precision independent of the order of the derivative which is in sharp contrast to methods of numerical differentiation.

2 Differential Algebras

In this section we will provide the mathematical background of the theory of differential algebras required for the advertised study of nonlinearities. Differential algebras are an offspring of the relatively new field of Nonstandard Analysis [8], [9], which allows the introduction of arbitrarily small quantities, "infinitesimals", in a rigorous theory of analysis. There is also some connection to the theories of formal power series [10] and automated differentiation [11]. The use of differential algebras for the field of nonlinear dynamics was first discussed in [12].

For the sake of clarity, we first address the simplest case of differential algebras, the structure ${}_1D_1$.

2.1 The Structure ${}_1D_1$

Consider the vector space R^2 of ordered pairs (a_0, a_1), $a_0, a_1 \in R$ in which an addition and a scalar multiplication are defined in the usual way:

$$(a_0, a_1) + (b_0, b_1) = (a_0 + b_0, a_1 + b_1) \tag{2}$$

$$t \cdot (a_0, a_1) = (t \cdot a_0, t \cdot a_1) \tag{3}$$

for $a_0, a_1, b_0, b_1 \in R$. Besides the above addition and scalar multiplication a multiplication between vectors is introduced in the following way:

$$(a_0, a_1) \cdot (b_0, b_1) = (a_0 \cdot b_0, a_0 \cdot b_1 + a_1 \cdot b_0) \tag{4}$$

for $a_0, a_1, b_0, b_1 \in R$. With this definition of a vector multiplication the set of ordered pairs becomes an algebra, denoted by ${}_1D_1$.

Note that the multiplication is the same one would obtain by multiplying $(a_0 + a_1 \cdot x)$ and $(b_0 + b_1 \cdot x)$ and keeping terms linear in x.

In the same way than in the case of complex numbers, one can identify $(a_0, 0)$ as the real number a_0. Where in the complex numbers, $(0, 1)$ was a root of -1, here it has another interesting property:

$$(0, 1) \cdot (0, 1) = (0, 0) \tag{5}$$

which follows directly from equation (4). So $(0, 1)$ is a root of 0. Such a property suggests thinking of $d = (0, 1)$ as something infinitely small, small enough that its square vanishes. Because of this we call $d = (0, 1)$ the differential unit. The first component of the pair (a_0, a_1) is called the real part, and the second component is called the differential part.

It is easy to verify that $(1, 0)$ is a neutral element of multiplication, because according to equation (4)

$$(1,0)\cdot(a_0,a_1)=(a_0,a_1)\cdot(1,0)=(a_0,a_1) \tag{6}$$

It turns out that (a_0,a_1) has a multiplicative inverse if and only if a_0 is nonzero; so ${}_1D_1$ is not a field. In case $a_0\neq 0$ the inverse is

$$(a_0,a_1)^{-1}=(\frac{1}{a_0},-\frac{a_1}{a_0^2}) \tag{7}$$

Using equations it is easy to check that in fact $(a_0,a_1)^{-1}\cdot(a_0,a_1)=(1,0)$.

The space ${}_1D_1$ is a subspace of the field R^* introduced in Nonstandard Analysis [8],[9]. Besides the usual real numbers, R^* contains a variety of infinitely small and infinitely large quantities. The outstanding result of the theory of Nonstandard Analysis is that differentiation becomes an algebraic problem: a function f is differentiable if and only if for any arbitrarily small quantity δ, the real part of the quotient

$$\frac{f(x+\delta)-f(x)}{\delta} \tag{8}$$

is independent of the choice of the specific δ. Thus, given any differentiable function f, we can compute its derivatives by just evaluating the formula for a special choice of δ. We choose $\delta=d=(0,1)$ and thus obtain

$$\begin{aligned} f'(x) &= \mathcal{R}[\frac{f(x+d)-f(x)}{d}] \quad \text{or} \\ f'(x) &= \mathcal{D}[f(x+d)-f(x)]=\mathcal{D}[f(x+d)] \end{aligned} \tag{9}$$

where $\mathcal{R}$ denotes the real part and $\mathcal{D}$ denotes the differential part. In the last step use has been made of the fact that $f(x)$ has no differential part. Hence differential algebras are useful to compute derivatives directly, without requiring an analytic formula for the derivative and without the inaccuracies of numerical techniques.

The computation of derivatives shall be illustrated in an example using the following function:

$$f(x)=\frac{1}{x+\frac{1}{x}} \tag{10}$$

The derivative of the function is

$$f'(x)=\frac{\frac{1}{x^2}-1}{(x+\frac{1}{x})^2} \tag{11}$$

Suppose we are interested in the value of the function and its derivative at x=2. We obtain

$$f(2) = \frac{2}{5} \quad f'(2) = -\frac{3}{25} \tag{12}$$

Now take the definition of the function f in equation (10) and evaluate it at $2 + d = (2, 1)$. One obtains:

$$\begin{aligned} f[(2,1)] &= \frac{1}{(2,1) + \frac{1}{(2,1)}} \\ &= \frac{1}{(2,1) + (\frac{1}{2}, -\frac{1}{4})} \\ &= \frac{1}{(\frac{5}{2}, \frac{3}{4})} \\ &= (\frac{2}{5}, -\frac{3}{4}/\frac{25}{4}) \\ &= (\frac{2}{5}, -\frac{3}{25}) \end{aligned} \tag{13}$$

As we can see, after the evaluation of the function the real part of the result is just the value of the function at $x = 2$, whereas the differential part is the derivative of the function at $x = 2$.

This is exactly what was to be expected from the theory of Nonstandard Analysis. However, for the sake of not relying on the quite advanced techniques of this relatively new field of mathematics, we also present an elementary but less elegant proof of the result.

By our choice of the starting vector $(2, 1)$, initially the vector contains the value $I(2)$ of the identity function $I : x \to x$ in the first component and the derivative of $I'(2) = 1$ in the second component.

Now assume that in an intermediate step two vectors of value and derivative $(g(2), g'(2))$ and $(h(2), h'(2))$ have to be added. According to (2) one obtains $(g(2) + h(2), g'(2) + h'(2))$. But according to the rule for the differentiation of sums, this is just the value and derivative of the sum function $(g + h)$ at $x = 2$.

The same holds for the multiplication: Suppose that two vectors of value and derivatives $(g(2), g'(2))$ and $(h(2), h'(2))$ have to be multiplied. Then according to (4) one obtains $(g(2) \cdot h(2), g(2) \cdot h'(2) + g'(2) \cdot h(2))$. But according to the product rule, this is just the value and derivative of the product function $(g \cdot h)$ at $x = 2$.

The evaluation of the function f at $(2, 1)$ can now be viewed as successively combining two intermediate functions g and h, starting with the identity function and finally arriving at f. At each intermediate step the derivative of the intermediate function is automatically obtained as the differential part according to the above reasoning.

An interesting side aspect is that with the search for a multiplicative inverse in equation (7) one has derived a rule to differentiate the function $f(x) = 1/x$ without explicitly using calculus rules.

After discussing the algebra ${}_1D_1$ and its virtues for the computation of derivatives, we now address the most general differential algebra, the structure ${}_nD_v$. It will eventually allow us to arithmetically compute partial derivatives of functions of v variables through order n.

2.2 The Structure ${}_nD_v$

We define $N(n,v)$ to be the number of monomials in v variables through order n. We will show that $N(n,v) = \frac{(n+v)!}{n!v!} = C(n+v,v)$ where $C(i,j)$ is the familiar binomial coefficient. First note that the number of monomials with **exact** order n equals $N(n,v-1)$. This is true because each monomial of exact order n can be written as a monomial with one variable less times the last variable to such a power that the total power equals n. Thus we have $N(n,v) = N(n-1,v) + N(n,v-1)$: the number of monomials in v variables through order n equals the number of monomials of one order less plus the ones of exact order n. This recursive relation is satisfied by $C(n+v,v)$. Since also obviously $C(1+1,1) = 2 = N(1,1)$, the formula follows by induction.

Now assume that all these N monomials are arranged in a certain manner order by order. For each monomial M we call I_M the position of M according to the ordering. Conversely, with M_I we denote the Ith monomial of the ordering. Finally, for an I with $M_I = x_1^{i_1} \cdot ... \cdot x_v^{i_v}$ we define $F_I = i_1! \cdot ... \cdot i_v!$.

We now define an addition, a scalar multiplication and a vector multiplication on R^N in the following way:

$$(a_1, ..., a_N) + (b_1, ...b_N) = (a_1 + b_1, ..., a_N + b_N) \tag{14}$$

$$t \cdot (a_1, ..., a_N) = (t \cdot a_1, ..., t \cdot a_N) \tag{15}$$

$$(a_1, ..., a_N) \cdot (b_1, ...b_N) = (c_1, ..., c_N) \tag{16}$$

where the coefficients c_i are defined as follows:

$$c_i = F_i \sum_{\substack{0 \le \nu, \mu \le N \\ M_\nu \cdot M_\mu = M_i}} \frac{a_\nu \cdot b_\mu}{F_\nu \cdot F_\mu} \tag{17}$$

To help clarify these definitions, let us look at the case of two variables and second order. In this case, we have $n = 2$ and $v = 2$. There are $N = C(2+2,2) = 6$ monomials in two variables, namely

$$1,\ x,\ y,\ xx,\ xy,\ yy \tag{18}$$

As an example, using the ordering in (18), we have $I_{xy} = 5$ and $M_3 = y$. Using the ordering in (18), we obtain for c_1 through c_6 in equation (17):

$$
\begin{aligned}
c_1 &= a_1 \cdot b_1 \\
c_2 &= a_1 \cdot b_2 + a_2 \cdot b_1 \\
c_3 &= a_1 \cdot b_3 + a_3 \cdot b_1 \\
c_4 &= 2 \cdot (a_1 \cdot b_4/2 + a_2 \cdot b_2 + a_4 \cdot b_1/2) \\
c_5 &= a_1 \cdot b_5 + a_2 \cdot b_3 + a_3 \cdot b_2 + a_5 \cdot b_1 \\
c_6 &= 2 \cdot (a_1 \cdot b_6/2 + a_3 \cdot b_3 + a_6 \cdot b_1/2)
\end{aligned} \tag{19}
$$

On ${}_nD_v$ we introduce a third operation ∂_i:

$$\partial_\nu(a_1, ..., a_N) = (c_1, ..., c_N) \tag{20}$$

with

$$c_i = \begin{cases} 0 & \text{if } M_i \text{ has order } n \\ a_{I_{(M_i \cdot x_\nu)}} & \text{else} \end{cases} \tag{21}$$

So ∂_ν moves the derivatives around in the vector. Suppose a vector contains the derivatives of the function f, then applying ∂_ν to it one obtains the derivatives of $\frac{\partial f}{\partial x_\nu}$ through one order less. With this third operation, ${}_nD_v$ becomes a Differential Algebra as defined in [12].

While in ${}_1D_1$, $d = (0, 1)$ was an infinitely small quantity, here we have a whole variety of infinitely small quantities that have the property that high enough powers of them vanish. We give special names to the ones in components I belonging to first order monomials, denoting them by dM_I. In the example of ${}_2D_2$, we have $dx = (0, 1, 0, 0, 0, 0)$ and $dy = (0, 0, 1, 0, 0, 0)$. It then follows from the theory of Nonstandard Analysis that instead of equation (9) we obtain

$$
\begin{aligned}
& f(x + dx, y + dy) = \\
& (f, \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial^2 f}{\partial x^2}, \frac{\partial^2 f}{\partial x \partial y}, \frac{\partial^2 f}{\partial y^2})(x, y)
\end{aligned} \tag{22}
$$

In the general case of v variables and order n, after evaluating f in the differential algebra one obtains:

$$\frac{\partial^{i_1 + i_2 + ... + i_v} f}{\partial x_1^{i_1} \partial x_2^{i_2} ... \partial x_v^{i_v}} = c_{I_{(x_1^{i_1} \cdot ... \cdot x_v^{i_v})}} \tag{23}$$

where $I_{(x_1^{i_1} \cdot ... \cdot x_v^{i_v})}$ is the index of the monomial $(x_1^{i_1} \cdot ... \cdot x_v^{i_v})$, as defined in the beginning of the section.

3 Important Functions on Differential Algebras

In this section we will generalize standard functions like exponentials, logarithmic and trigonometric function to differential algebra. As we will see below, virtually all functions existing on a computer can be generalized in a straightforward way.

We start our discussion by noting that for any differential algebra vector of the form $(0, a_1, ..., a_N) \in {}_nD_v$, i.e. with a zero in the component belonging to the zeroth order monomial, we have the following property:

$$(0, a_1, ..., a_N)^i = (0, 0, ..., 0) \quad \text{for } i > n \tag{24}$$

which follows directly from the definition of the multiplication in ${}_nD_v$ defined in equation (16).

Let us begin our discussion of special functions with the exponential function $\exp(x)$. Assume we have to compute the exponential of a differential algebra vector that has already been created by previous operations. First we note that the functional equation $\exp(x+y) = \exp(x) \cdot \exp(y)$ also holds in Nonstandard Analysis. As we will see, this facilitates the computation of the exponential considerably.

$$\begin{aligned}
\exp[(a_0, a_1, a_2, ..., a_N)] &= \exp(a_0) \cdot \exp[(0, a_1, a_2, ..., a_N)] \\
&= \exp(a_0) \cdot \sum_{i=0}^{\infty} \frac{(0, a_1, a_2, ..., a_N)^i}{i!} \\
&= \exp(a_0) \cdot \sum_{i=0}^{n} \frac{(0, a_1, a_2, ..., a_N)^i}{i!}
\end{aligned} \tag{25}$$

In the last step use has been made of equation (24) which entails that the sum has to be taken only through order n and thus allows the computation of the root in finitely many steps. Hence the evaluation of the real number exponential $\exp(a_0)$ which internally on a computer requires a power series summation and hence cannot be done accurately, is more subtle then the rest of the operations in differential algebra.

A logarithm of a differential algebra vector exists if and only if $a_0 > 0$. In this case one obtains

$$\begin{aligned}
\log[(a_0, a_1, a_2, ..., a_N)] &= \log[a_0 \cdot (1 + (0, \frac{a_1}{a_0}, \frac{a_2}{a_0}, ... \frac{a_N}{a_0})] \\
&= (\log(a_0), 0, ..., 0) + \sum_{i=1}^{\infty} (-1)^{i+1} \frac{1}{i} (0, \frac{a_1}{a_0}, \frac{a_2}{a_0}, ..., \frac{a_N}{a_0})^i \\
&= (\log(a_0), 0, ..., 0) + \sum_{i=1}^{n} (-1)^{i+1} \frac{1}{i} (0, \frac{a_1}{a_0}, \frac{a_2}{a_0}, ..., \frac{a_N}{a_0})^i
\end{aligned}$$

$$(26)$$

Again use has been made of the fundamental property of the logarithm $\log(x \cdot y) = \log(x) + \log(y)$ which transforms directly into Nonstandard Analysis and leads to simplifications by virtue of equation (24).

As the last example, we will derive a formula for the root function. Even though there is a direct method to compute roots by solving a set of linear equations for the coefficients of the root, we present here a technique based on power series following an approach similar to the exponential and logarithm. The root has the following power series expansion:

$$\sqrt{1+x} = \sum_{i=0}^{\infty} (-1)^i \frac{1 \cdot 3 \cdot \ldots \cdot (2i-3)}{2 \cdot 4 \cdot \ldots \cdot (2i)} \cdot x^i \tag{27}$$

Using this formula and the definitions of addition and multiplication (14), (16), one directly obtains for the square root of a differential algebra vector:

$$\begin{aligned}
& \sqrt{(a_0, a_1, a_2, \ldots, a_N)} \\
= \; & \sqrt{a_0} \cdot \sqrt{1 + (0, \frac{a_1}{a_0}, \frac{a_2}{a_0}, \ldots, \frac{a_N}{a_0})} \\
= \; & \sqrt{a_0} \cdot \sum_{i=0}^{\infty} (-1)^i \frac{1 \cdot 3 \cdot \ldots \cdot (2i-3)}{2 \cdot 4 \cdot \ldots \cdot (2i)} \cdot (0, \frac{a_1}{a_0}, \frac{a_2}{a_0}, \ldots, \frac{a_N}{a_0})^i \\
= \; & \sqrt{a_0} \cdot \sum_{i=0}^{n} (-1)^i \frac{1 \cdot 3 \cdot \ldots \cdot (2i-3)}{2 \cdot 4 \cdot \ldots \cdot (2i)} \cdot (0, \frac{a_1}{a_0}, \frac{a_2}{a_0}, \ldots, \frac{a_N}{a_0})^i
\end{aligned} \tag{28}$$

Using the addition theorems for sine and cosine, one obtains formulas with finite sums in a quite similar way; in general, suppose a function f has an addition theorem of the form

$$f(a+b) = g_a(b) \tag{29}$$

and $g_a(b)$ can be written in a power series, then by the same reasoning its differential algebraic extension is computable exactly in only finitely many steps. In practice it turns out that this can be done for all commonly supported functions in a FORTRAN computer environment.

4 The Implementation of Differential Algebra on a Computer

The arithmetic and the functions of differential algebra can be implemented on a computer for arbitrary order and arbitrary number of variables. As it turns out, this is not easily possible for the field R^* of Nonstandard Analysis; hence we sacrifice the universal existence of multiplicative inverses.

The implementation of the addition and scalar multiplication is trivial. However, the efficient implementation of a multiplication requires some care. First we note that we can increase the speed by defining a slightly different multiplication than in equation (17) with the c_i as

$$c_i = \sum_{\substack{0 \le \nu, \mu \le N \\ M_\nu \cdot M_\mu = M_i}} a_\nu \cdot b_\mu \tag{30}$$

This multiplication is the same as in the case of the multiplication of power series. However, in this arithmetic not all power series coefficients of the product are computed; coefficients belonging to terms of orders higher than n are disregarded. For many cases this view of differential algebras as "truncated power series algebras" (see ref. [13]) is sufficient.

The definition of the multiplication (16) requires the knowledge of all possible factorizations of a monomial into two submonomials. The computation of all these factorizations can be quite time consuming. Additionally, in practice it happens frequently that many of the entries in a differential algebra vector are zero.

So it is advantageous to turn the problem inside out so that no factorizations in submonomials are searched, but rather each component of the first vector is multiplied by each component of the second vector and the product is stored at the place where the product monomial belongs.

In order to do that an easy way of finding the address of the product monomial is required. This is done as follows. First, all $N(n, v)$ monomials M are coded with an integer C in the following way: Let $M = x_1^{i_1} \cdot ... \cdot x_v^{i_v}$ Then we define $C(M)$ as follows:

$$C(M) = C(x_1^{i_1} \cdot ... \cdot x_v^{i_v}) = i_1 \cdot (n+1)^0 + i_2 \cdot (n+1)^1 + ... + i_v \cdot (n+1)^{(v-1)} \tag{31}$$

This means that the exponents are just "decimals" in base $(n+1)$. Note that since $i_\nu \le n$ this representation is injective, i.e. different monomials have different codings. Note also that all codings are always less than $(n+1)^v$, but not all such codings occur.

Now suppose two monomials M and N have to be multiplied and suppose their product has an order less than or equal to v. Since the multiplication corresponds to an addition of the exponents, it follows that

$$C(M \cdot N) = C(M) + C(N) \tag{32}$$

To exploit this for the finding of the desired coordinate position I_M (see section 2) of the product of two monomials, an array D is required that has the property

$$I_M = D(C(M)) \tag{33}$$

This array can be generated easily by the computer. Since the codings are bounded by $(n+1)^v$, the array has to have at least this length. With 6 variables, this enables orders of 8 or 9 if one wants to stay inside the boundaries of computer storage; with 8 variables the order would decrease to about 4 which is too strict a limitation. To circumvent this, a slight modification of the above coding and decoding will be presented.

Without loss of generality, we assume the number of variables v to be even; if it is not even, increase it by one and ignore the additional variable. We define two coding numbers C_1 and C_2 for any monomial in the following way:

$$\begin{aligned} C_1(x_1^{i_1} \cdot ... \cdot x_v^{i_v}) &= i_1 \cdot (n+1)^0 + i_2 \cdot (n+1)^1 + ... + i_{\frac{v}{2}} \cdot (n+1)^{(\frac{v}{2}-1)} \\ C_2(x_1^{i_1} \cdot ... \cdot x_v^{i_v}) &= i_{\frac{v}{2}+1} \cdot (n+1)^0 + i_{\frac{v}{2}+2} \cdot (n+1)^1 + ... + i_v \cdot (n+1)^{(\frac{v}{2}-1)} \end{aligned} \tag{34}$$

Then we store the $N(n,v)$ monomials in the following way. We start with all monomials that have $C_2(M) = 0$ and group them by order; within one order, the monomials are stored according to ascending values of $C_1(M)$. Then we store all those with $C_2(M) = 1$, again by order, and so forth, going through all possible values of C_2. Again we obtain

$$\begin{aligned} C_1(M \cdot N) &= C_1(M) + C_1(N) \\ C_2(M \cdot N) &= C_2(M) + C_2(N) \end{aligned} \tag{35}$$

Finally we introduce some "inverse" arrays D_1 and D_2 in the following way:

$$\begin{aligned} D_1(c_1) &= (I_M \text{ of first monomial } M \text{ with } C_1(M) = c_1) \\ D_2(c_2) &= (I_M \text{ of first monomial } M \text{ with } C_2(M) = c_2) - 1 \end{aligned} \tag{36}$$

Again the arrays D_1 and D_2 can be generated by the computer.

Using the definitions of C_1, C_2, D_1 and D_2 and using the storage scheme outlined above, it now follows that the address of the product of the monomials M and N can be found directly as

$$I_{M \cdot N} = D_1[C_1(I_M) + C_1(I_N)] + D_2[C_2(I_M) + C_2(I_N)] \tag{37}$$

For the sake of clarity, an example for the arrays c_1, c_2, d_1 and d_2 is given in appendix 1 for $n = 3$ and $v = 4$. This example also illustrates equations (34) through (37).

The coding defined in (34) entails that the maximum length of the arrays D_1 and D_2 can now be chosen much lower, namely as $(n+1)^{\frac{v}{2}}$. Assuming a maximum length of 1 million, this entails limitations on the maximum order given a certain number of variables as listed in table 1.

Table 1: The maximum order for different numbers of variables due to a limitation of the length of the reverse addressing arrays D_1, D_2

number of variables	6	8	10	12
maximum order	99	30	14	10

After addition and multiplication are available, the implementation of differential algebra functions is done quite easily using the formulas discussed in section 3.

For practical purposes it is of importance that in the FORTRAN environment differential algebraic operations can only be utilized by calls to subroutines. For this reason a precompiler [14] was developed that allows the use of a new data type "differential algebra" in regular FORTRAN formulas. The precompiler parses the entire program and transforms formulas containing differential algebraic quantities into subroutine calls.

5 The Computation of Transfer Maps

5.1 An Illustrating Example

Differential Algebras can be used very efficiently to compute the transfer map of equation (1) of particle optical systems in its Taylor series representation.

To illustrate this, let us start the discussion with a very simple example, the midplane motion in a 90^o homogenous bending magnet. Let x_i and $a_i = \sin(\alpha_i)$ denote the initial distance and scaled transverse momentum relative to the reference trajectory (see Fig. 1). Then we are interested in the values x_f and $a_f = \sin(\alpha_f)$. Since the trajectories in the magnet are circles, we can readily read from Fig. 1:

$$\begin{aligned} A &= R\ \sin(\alpha_i) = R\ a_i \\ B &= R\ (1-\cos(\alpha_i)) + x_i = R\ (1-\sqrt{1-a_i^2}) + x_i \\ a_f &= \sin(\alpha_f) = -\frac{B}{R} \end{aligned}$$

Figure 1: The motion in a 90^o homogeneous bending magnet with radius of bend R

$$x_f \;=\; A - R\,(1-\cos(\alpha_f)) = A - R\,(1-\sqrt{1-a_f^2}) \tag{38}$$

These equations allow the computation of the final coordinates x_f, a_f in terms of the initial coordinates x_i, a_i. However, taking these equations and performing all operations in differential algebra allows us to even obtain all derivatives of $x_f,\ a_f$ with respect to $x_i,\ a_i$. These so obtained derivatives, evaluated at $x_i = 0,\ a_i = 0$, are then the expansion coefficients of the map in equation (1). For the sake of clarity, let us explicitly show how x_f and a_f are computed.

Using the ordering in (18) and identifying the variable a with y, we obtain using the arithmetic defined in equations (14), (15) and (16)

$$\begin{aligned}
x_i &= (0,1,0,0,0,0)\\
a_i &= (0,0,1,0,0,0)\\
A &= (0,0,R,0,0,0)\\
B &= (0,1,0,0,0,R)\\
a_f &= (0,-\frac{1}{R},0,0,0,-1)\\
x_f &= (0,0,R,-\frac{1}{R},0,0)
\end{aligned} \tag{39}$$

Comparing the so obtained result with any matrix code [1], [2], [3], [5], we find complete agreement; as an example, the fact that the second component of x_f is zero implies that $\frac{\partial x_f}{\partial x_i} = 0$ and hence (x,x) $= 0$ (or in TRANSPORT notation $\mathrm{R}_{1,1} = 0$) which is a well known property of 90^o bends.

In case an additional particle optical element is to follow this bending magnet, one does not have to start all over evaluating this new element at $x_i = (0,1,0,0,0,0)$, $a_i = (0,0,1,0,0,0)$, but one can start already with x_f and a_f of equation (39). This way one can save the usually quite involved concatenation process and increase performance significantly.

The example discussed in this section has been implemented on the computer. Using the Differential Algebra package, it is easily possible to extract all nonlinearities of this two-dimensional example through order fifty.

5.2 Generation of Maps Using Numerical Integration

In this section we will address the general case in which no closed solution of the problem exists. We will see that also in this case we are actually able to compute transfer maps of arbitrary order for arbitrary particle optical elements. Even though we do not have analytical formulas that relate the final coordinates to the initial coordinates, there is still a way to computationally relate the final coordinates to the initial coordinates, by numerical integration of the equations of motion.

In this case, the final coordinates are still computed from the initial coordinates using standard arithmetic and functions, however the relations are more complex than in the case of the homogeneous sector.

Now blindfoldedly performing all these operations in differential algebra automatically leads to all desired derivatives of the transfer function, regardless of the form of the equations of motion.

Differential algebraic techniques have been implemented in the program COSY [5]. They allow the computation of transfer maps of elements with a dependence on the independent variable for which an analytic solution cannot be obtained from HAMILTON [4] like fringing fields. Using an eighth order Runge Kutta integrator, all operations required for a tracking of particles are performed in differential algebra. This allows the computation of arbitrary fringing field effects as soon as the spacial distribution of the electromagnetic fields is known [15].

In many cases, including the proposed SSC [18], the particle optical system can be represented very well by a sequence of kicks as in TEAPOT [16]. The sequence of kicks can be viewed as a symplectic integrator of second order in the independent variable. In this case, the execution of all arithmetic in differential algebra is particularly easy. It may be used for very efficient and accurate computation of tune shifts and chromaticities using normal form theory [6], [7]. This will be discussed in detail in [17].

It is worth mentioning that TEAPOT and COSY are in complete agreement for all nonlinearities through fifth order. In order to show this, we selected a small ring for which bends and quadrupoles of finite length had to be split into several hundred drifts and kicks in order to be computable by TEAPOT. For the actual example and the reason for the ring selection see reference (15).

5.3 Hamiltonian Theory

In this subsection we will outline the usefulness of differential algebras for Hamiltonian systems. One of the most fundamental concepts of Hamiltonian theory is the Poisson bracket between two functions of phase space. This requires the differentiation with respect to phase space variables.

Suppose a differential algebra vector is given. Then it can be viewed as a descriptor of a function, giving its value and derivatives at a certain point. In this context the required differentiation is just the "bookkeeping operation" moving derivatives in the vector to different places introduced in section (2.2). Thus a Poisson bracket for arbitrary order and arbitrary number of variables can be computed.

Using the Poisson bracket, Lie operators, "Poisson brackets waiting to happen", can be computed. In fact, as soon as the generator $\vec{v}$ of the Lie operator $:\vec{v}:$ vanishes at the origin and has zero first derivatives, the process is closed in that no feeddown from higher orders occurs.

Using Lie operators, the transfer map or flow of a time independent Hamiltonian system can be computed as

$$\mathcal{M} = \exp(-t :H:) \tag{40}$$

where H is the Hamiltonian of the system. Note that by the proper choice of the coordinates it can always be achieved that $H(\vec{0}) = 0$ and also the first derivatives of H vanish. This entails that each summand in equation (40) can be computed in a closed fashion.

Furthermore, it turns out it suffices to carry only finitely many terms of the sum in (40) if we want the nth order coefficients of the transfer map to a certain accuracy. To see this, we show that the norm of the Lie operator $:-tH:$ is bounded. First note that in order to obtain the transfer map through order n, it suffices to know H to order $n+1$.

We define a norm on Differential Algebra vectors in the following way

$$|(a_1, ..., a_N)| = \sum_{i=1}^{N} |a_i| \tag{41}$$

It is quite easy to show that for arbitrary vectors V_1, V_2 we have $|V_1 \cdot V_2| \leq |V_1| \cdot |V_2|$. Since the "derivative" ∂_ν defined in section (2.2) only moves coefficients around in the vector and drops some, it follows for a vector V that

$$|\partial_\nu V| \leq |V| \tag{42}$$

and thus we obtain for the norm of the Lie operator consisting of the two derivatives in each summand:

$$\begin{aligned} |:-tH:| &= \sup \frac{|-t:H:V|}{|V|} \\ &= \sup \frac{\sum_{i=1}^{v/2} |t| \cdot \big((\partial_i H)(\partial_{i+v/2} V) - (\partial_i V)(\partial_{i+v/2} H)\big)}{|V|} \\ &\leq |tH| \cdot v \end{aligned} \tag{43}$$

This entails that the sum $\sum_{\nu=0}^{\infty} \frac{(-t:H:)^\nu}{\nu!}$ converges (in a differential algebra sense!), and that it is possible to obtain an estimate as to how many terms have to be taken in order to obtain a certain accuracy. In general it turns out that these estimates are even usually quite pessimistic and the terms in the sum become small more quickly.

This method to obtain the Hamiltonian flow in a computer code has the additional advantage that it does not require any composition of the maps of two subsystems; all that is required is to compute H by substituting in the $\vec{z}$ already obtained and then let this differential algebraic vector act on $\vec{z}$.

In practice it turns out that the savings from the avoided compositions is more significant than the additional time required for the iterative evaluation of the exponential sum $\exp(:-tH:)$. In fact, a composition routine requires $N(n,v)$ differential algebra multiplications, whereas one term in the sum requires v multiplications per dimension.

In the case the Hamiltonian is time dependent, the above reasoning requires a slight modification. In this case it is in general not possible to go through the time dependent element in one step. However, it is possible to derive an explicit high order numerical integrator in the following way. First we note that for any function of phase space, we have

$$\frac{df}{dt} = [f,H] + \frac{\partial}{\partial t} f = \mathcal{O} f \tag{44}$$

In our particular case, we are especially interested in the cases in which f are the components of $\vec{z}$. Iterating equation (44), we can obtain higher order derivatives of f, say up to order u. First note that even for the f's of interest in, $\frac{\partial}{\partial t} f$ vanishes, but in the higher derivatives such partial derivatives prevail because H is time dependent.

In order to guarantee that all derivatives of the phase space variables can be computed to order u, H must be known to order $v+u$. However, terms with a higher power of t than u can be set to zero. In order to compute the u derivatives of the phase space vector, which is needed for the uth order integrator, a total of $6 \cdot u$ Poisson brackets are required. Considering that in many practical cases, an element can be transversed in one or very few steps of an eighth order integrator and that like above we still save the composition process, the effort is still very favorable to the situations in which compositions are required.

Acknowledgements

For financial support I want to thank the Deutsche Forschungsgemeinschaft, Dr. Alex Chao of the SSC Central Design Group and Dr. Edward Heighway of Los Alamos National Laboratory. For fruitful discussions I want to thank Dr. Etienne Forest and Professor Dr. Hermann Wollnik.

6 References

1. K.L Brown, D. C. Carey, Ch. Iselin and F. Rothacker, SLAC 91 (1973 rev.), NAL 91, and CERN 80-04

2. A.J. Dragt et. al., IEEE Trans. Nucl. Sci., NS-32, 2311 (1985)

3. H. Wollnik, J. Brezina, M. Berz and W. Wendel, Proc. AMCO-7, GSI-Rep. THD-26 (1984) p.679

4. M. Berz and H. Wollnik, Nucl. Instr. Meth. A258 (1987) 364

5. M. Berz, H. Wollnik and H. C. Hoffmann, Nucl. Instr. Meth. A 258 (1987) 402

6. E. Forest, Particle Accelerators 22 (1987) 15

7. E. Forest, SSC Report SSC-111, Berkeley

8. A. Robinson, Proc. Royal Acad. Amsterdam Ser. A, 64, 432-440, 1961

9. D. Laugwitz, JBer. Deutsch. Math. Vereinigung 75 (1973) 66

10. I. Niven, Am. Math. Monthly 76-8 (1969) 871

11. L. B. Rall, Math. Magazine 59 (1986) 275

12. J. F. Ritt, "Differential Algebra", American Mathematical Society, 1950

13. M. Berz, Nucl. Instr. Meth. A258 (1987) 431 ; M. Berz, Los Alamos Accelerator Theory Note AT-6:ATN-86-16 (1986)

14. M. Berz, Los Alamos Report AT-3:TN-87-32 (1987)

15. M. Berz, Dissertation University of Giessen, 1986 (unpublished, in German); B. Hartmann, Diplomarbeit University of Giessen 1987 (unpublished, in German)

16. L. Schachinger and R. Talman, Particle Accelerators 22 (1987) 35

17. E. Forest, M. Berz and J. Irwin, this issue of Particle Accelerators

18. SSC Conceptual Design Report, SSC Note SSC-SR 2020, Berkeley

7 Appendix

List of the ordering of the all monomials $M = x_1^{i_1} \cdot ... \cdot x_v^{i_v}$ for order $n = 3$ and number of variables $v = 4$. Also shown are the coding integers C_1 and C_2 (cf section 4)

I_M	i_1	i_2	i_3	i_4	C_1	C_2
1	0	0	0	0	0	0
2	1	0	0	0	1	0
3	0	1	0	0	4	0
4	2	0	0	0	2	0
5	1	1	0	0	5	0
6	0	2	0	0	8	0
7	3	0	0	0	3	0
8	2	1	0	0	6	0
9	1	2	0	0	9	0
10	0	3	0	0	12	0
11	0	0	1	0	0	1
12	1	0	1	0	1	1
13	0	1	1	0	4	1
14	2	0	1	0	2	1
15	1	1	1	0	5	1
16	0	2	1	0	8	1
17	0	0	0	1	0	4
18	1	0	0	1	1	4
19	0	1	0	1	4	4
20	2	0	0	1	2	4
21	1	1	0	1	5	4
22	0	2	0	1	8	4
23	0	0	2	0	0	2
24	1	0	2	0	1	2
25	0	1	2	0	4	2
26	0	0	1	1	0	5
27	1	0	1	1	1	5
28	0	1	1	1	4	5
29	0	0	0	2	0	8
30	1	0	0	2	1	8
31	0	1	0	2	4	8
32	0	0	3	0	0	3
33	0	0	2	1	0	6
34	0	0	1	2	0	9
35	0	0	0	3	0	12

List of the arrays D_1 and D_2 for order $n = 3$ and number of variables $v = 4$. For all M, one has $I_M = D_1(C_1(M)) + D_2(C_2(M))$ (cf section 4)

j	$D_1(j)$	$D_2(j)$
0	1	0
1	2	10
2	4	22
3	7	31
4	3	16
5	5	25
6	8	32
7	0	0
8	6	28
9	9	33
10	0	0
11	0	0
12	10	34